Отчёт по лабораторной работе №8

Дисциплина: архитектура компьютеров и операционные системы

Зарина Исмайилбековна Исаева

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

Приобретение навыков написания программ с использованием циклов и обработкой аргументов командной строки.

2 Задание

1. Реализация циклов в NASM.
2. Обработка аргументов командной строки.
3. Задание для самостоятельной работы.

3 Теоретическое введение

Стек — это структура данных, организованная по принципу LIFO («Last In — First Out» или «последним пришёл — первым ушёл»). Стек является частью архитектуры процессора и реализован на аппаратном уровне. Для работы со стеком в процессоре есть специальные регистры (ss, bp, sp) и команды. Основной функцией стека является функция сохранения адресов возврата и передачи аргументов при вызове процедур. Кроме того, в нём выделяется память для локальных переменных и могут временно храниться значения регистров. Стек имеет вершину, адрес последнего добавленного элемента, который хранится в регистре esp (указатель стека). Противоположный конец стека называется дном. Значение, помещённое в стек последним, извлекается первым. При помещении значения в стек указатель стека уменьшается, а при извлечении — увеличивается.

Команда push размещает значение в стеке, т.е. помещает значение в ячейку памяти, на которую указывает регистр esp, после этого значение регистра esp увеличивается на 4. Данная команда имеет один операнд — значение, которое необходимо поместить в стек.

Команда pop извлекает значение из стека, т.е. извлекает значение из ячейки памяти, на которую указывает регистр esp, после этого уменьшает значение регистра esp на 4. У этой команды также один операнд, который может быть регистром или переменной в памяти. Нужно помнить, что извлечённый из стека элемент не стирается из памяти и остаётся как “мусор”, который будет перезаписан при записи нового значения в стек.

Для организации циклов существуют специальные инструкции. Для всех ин-

струкций максимальное количество проходов задаётся в регистре ecx. Наиболее простой является инструкция loop. Она позволяет организовать безусловный цикл.

4 Выполнение лабораторной работы

4.1 Реализация циклов в NASM

Создаю каталог для программ лабораторной работы № 8, перехожу в него и создаю файл lab8-1.asm (рис. [4.1]).

Создание файлов для лабораторной работы

Рис. 4.1: Создание файлов для лабораторной работы

Ввожу в файл lab8-1.asm текст программы из листинга 8.1 (рис. [4.2]).

Ввод текста программы из листинга 8.1

Рис. 4.2: Ввод текста программы из листинга 8.1

Создаю исполняемый файл и запускаю его (рис. [4.3]).

Запуск программного кода

Рис. 4.3: Запуск программного кода

Данная программа выводит числа от N до 1 включительно.

Изменяю текст программы, добавив изменение значения регистра ecx в цикле (рис. [4.4]).

Редактирование текста программы

Рис. 4.4: Редактирование текста программы

Создаю исполняемый файл и проверяю его работу (рис. [4.5])

Запуск обновлённой программы

Рис. 4.5: Запуск обновлённой программы

В данном случае число проходов цикла не соответствует введенному с клавиатуры значению.

Вношу изменения в текст программы, добавив команды push и pop для сохранения значения счетчика цикла loop (рис. [4.6]).

Редактирование текста программы

Рис. 4.6: Редактирование текста программы

Создаю исполняемый файл и проверяю его работу (рис. [4.7])

Вывод программы

Рис. 4.7: Вывод программы

В данном случае число проходов цикла соответствует введенному с клавиатуры значению и выводит числа от N-1 до 0 включительно.

4.2 Обработка аргументов командной строки

Создаю файл lab8-2.asm в каталоге ~/work/arch-pc/lab07 (рис. [4.8]).

Ввод текста из листинга 8.2

Рис. 4.8: Ввод текста из листинга 8.2

Создаю исполняемый файл и запускаю его, указав нужные аргументы (рис. [4.9]).

Запуск исполняемого файла

Рис. 4.9: Запуск исполняемого файла

Программа вывела 4 аргумента, так как аргумент 2 не взят в кавычки, в отличии от аргумента 3, поэтому из-за пробела программа считывает “2” как отдельный аргумент.

Рассмотрим пример программы, которая выводит сумму чисел, которые передаются в программу как аргументы. Создаю файл lab8-3.asm в каталоге ~/work/archpc/lab08 и ввожу в него текст программы из листинга 8.3 (рис. [4.10])

Ввод текста из листинга 8.3

Рис. 4.10: Ввод текста из листинга 8.3

Создаю исполняемый файл и запускаю его, указав аргументы (рис. [4.11]).

Вывод программы

Рис. 4.11: Вывод программы

Изменяю текст программы из листинга 8.3 для вычисления произведения аргументов командной строки (рис. [4.12])

Изменение текста файла

Рис. 4.12: Изменение текста файла

Создаю исполняемый файл и запускаю его, указав аргументы (рис. [4.13]).

Запуск исполняемого файла

Рис. 4.13: Запуск исполняемого файла

4.3 Задания для самостоятельной работы

Пишу текст программы, которая находит сумму значений функции f(x) = 10x - 5 в соответствии с моим номером варианта (3) для x = x1, x2, ..., xn. Значения xi передаются как аргументы. (рис. [4.14])

Текст программы

Рис. 4.14: Текст программы

Создаю исполняемый файл и проверьте его работу на нескольких наборах x = x1, x2, ..., xn. (рис. [4.15]).

Запуск исполняемого файла и его проверка

Рис. 4.15: Запуск исполняемого файла и его проверка

Программа работает корректно.

Код программы:

%include ‘in_out.asm’ SECTION .data esult_msg db “Результат:”, 0 coeff dd 10 constant dd 5 SECTION .text global _start

_start: pop ecx sub ecx, 1 mov esi, 0 next_value: cmp ecx, 0 je print_result pop ebx call atoi eax, [coeff] sub eax, [constant] add esi, eax loop next_value print_result: mov eax, result_msg call sprint mov eax, esi call iprintLF call quit

5 Выводы

Благодаря данной лабораторной работе я приобрела навыки написания программ использованием циклов и обработкой аргументов командной строки, что поможет мне при выполнении последующих лабораторных работ.

Список литературы

1. [Лабораторная работа №8] (https://esystem.rudn.ru/pluginfile.php/2089095/mod_resource/co